Self-Consistent Learning of Neural Dynamical Systems From Noisy Time Series

Zhe Wang and Claude Guet

***Abstract*—We introduce a new method which, for a single noisy time series, provides unsupervised filtering, state space reconstruction, efficient learning of the unknown governing multivariate dynamical system, and deterministic forecasting. We construct both the underlying trajectories and a latent dynamical system using deep neural networks. Under the assumption that the trajectories follow the latent dynamical system, we determine the unknowns of the dynamical system, and filter out stochastic outliers in the measurements. In this sense the method is self-consistent. The embedding dimension is determined iteratively during the training by using the false-nearest-neighbors Algorithm and it is implemented as an attention map to the state vector. This allows for a state space reconstruction without a priori information on the signal. By exploiting the differentiability of the neural solution trajectory, we can define the neural dynamical system locally at each time, mitigating the need for forward and backwards passing through numerical solvers of the canonical adjoint method. On a chaotic time series masked by additive Gaussian noise, we demonstrate that the denoising ability and the predictive power of the proposed method are mainly due to the self-consistency, insensitive to methods used for the state space reconstruction.**

***Index Terms*—Denoising and deterministic forecasting, neural dynamical systems, self-consistent learning.**

I. Introduction

TIME series of measured data have been studied to extract information on the unknown underlying dynamical systems, and to predict the future of observables from past measurements. The first systematic study dates back to 1927 when Yule introduced a linear autoregression model to reveal the dynamics of sunspot numbers [1]. The model which writes as

$$u(k+1) = \sum_{i=0}^{m-1} a(i)u(k-i) + e(k), \tag{1}$$

states that the future $u(k+1)$ is a weighted sum of the past values in the sequence by the coefficients $a(i)$, with the error term $e(k)$. Here, m denotes the regression order.

Since linear equations lead to only exponential or periodic motions, systems characterized by irregular motions call for nonlinear models which advantageously employ neural networks. Retaining the basic form of (1), neural autoregression models can be expressed as

$$u(k+1) = \mathcal{N}_{AR}[\tilde{\boldsymbol{u}}(k); \boldsymbol{\theta}_{AR}] + e(k), \tag{2}$$

where $\tilde{\boldsymbol{u}}(k) = [u(k), \ldots, u(k-(m-1))]^T$ denotes an m-dimensional delay vector. The parameters $\boldsymbol{\theta}_{AR}$ of the neural network, denoted by $\mathcal{N}_{AR}(\cdot) : \mathbb{R}^m \to \mathbb{R}$, are determined by minimizing the deviation $e(k)$. Since the first application of neural networks to model a chaotic time series in [2], various network architectures including multilayer perceptrons [3], [4], time delayed neural networks [5], [6], convolution neural networks [7], recurrent neural networks [8]–[10], and more recently transformers [11], [12], have been explored to study time series arising from physics, engineering, biological science, finance, etc., cf. [13] for a survey.

Using neural networks as a nonlinear autoregression model stems from the universal approximation theorem which states that a sufficiently deep neural network can approximate any arbitrary well-behaved nonlinear function with a finite set of parameters [14]–[16] and from the Takens' theorem [17] in the following manner. Let $\boldsymbol{v}(t)$ be a state vector and let

$$\frac{\mathrm{d}\boldsymbol{v}(t)}{\mathrm{d}t} = \boldsymbol{f}[\boldsymbol{v}(t)], \tag{3}$$

be the governing equation. One seldom has access to $\boldsymbol{v}(t)$. Instead, the state of dynamical systems is partially observed and inferred through a sequence of scalar measurements $u(k)$ sampled at discrete times and, frequently, masked by noise. For a sufficiently large dimension $m \in \mathbb{Z}^+$ and an arbitrary delay time $\tau \in \mathbb{R}^+$, Takens' theorem [17] asserts the existence of a diffeomorphism between the corresponding delay vector $\tilde{\boldsymbol{u}}(k)$ and the underlying state $\boldsymbol{v}(t)$ of the dynamical system. This implies that there exists a nonlinear mapping $u(k+1) = g[\tilde{\boldsymbol{u}}(k)]$ which models the time series exactly. In virtue of the universal approximation theorem, neural networks could capture the $g(\cdot)$ mapping.

State space reconstruction with the time delayed vector was first proposed by Ruelle (cf. [18]) and advocated by Packard *et al.* [19], and later proved by Takens [17]. It turns out that

Manuscript received August 12, 2021; revised October 15, 2021 and November 15, 2021; accepted December 6, 2021. The work of Zhe Wang was supported by Energy Research Institute@NTU, Nanyang Technological University, where most of this work was performed. This work was supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme. *(Corresponding author: Zhe Wang.)*

Zhe Wang is with CNRS@CREATE, Singapore 138602 (e-mail: zhe.wang@cnrsatcreate.sg).

Claude Guet is with Energy Research Institute @NTU, Nanyang Technological University, Singapore 637141, and also with the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore 637371 (e-mail: cguet@ntu.edu.sg).

This article has supplementary downloadable material available at https://doi.org/10.1109/TETCI.2022.3146332, provided by the authors.

Digital Object Identifier 10.1109/TETCI.2022.3146332

delay vectors of sufficient length are not only a representation of the state of a linear system (1), but, more profoundly, able to recover the full geometric property of nonlinear dynamical systems. Measured time series are often corrupted by noises. In order to remove noise, Sauer [20] applied a low-pass filter to the delayed vector space, from which a local linear model (1) is constructed. Going beyond the delay embedding, recent studies on deep state space reconstruction [21], [22] have shown that the denoising feature can be achieved using an autoencoder. In these studies, the state space reconstruction was performed with a prescribed value of the embedding dimension, which, however, was not known a priori. In view of this deficiency, Gilpin [23] incorporated the false-nearest-neighbor (FNN) Algorithm [24] into a loss function to penalize the encoder outputs. As a consequence, the reconstructed attractor is confined to a subspace smaller than the configuration space. The major drawback of such an approach is that the FNN-regularizer penalizes not only the redundant dimensions but also the subspace that contains the attractor. As a consequence, the results of the state space reconstruction depends sensitively on the strength of the regularizer.

Recently, Ouala *et al.* [22] proposed to reconstruct a latent difference equation from the time series

$$\boldsymbol{u}(k+1) = \mathcal{N}_{DE}[\boldsymbol{u}(k); \boldsymbol{\theta}_{DE}] + \boldsymbol{e}(k). \tag{4}$$

With a prescribed value of m, the m-dimensional state vector $\boldsymbol{u}(k)$ is obtained by embedding the measured time series to the latent space of an encoder. A joint minimization of the vector-valued one-step prediction error $\boldsymbol{e}(k)$ in addition to the reconstruction loss of the autoencoder gave simultaneously the autoencoder and the parameters $\boldsymbol{\theta}_{DE}$ of the map $\mathcal{N}_{DE}(\cdot) : \mathbb{R}^m \to \mathbb{R}^m$. Compared with the neural autoregression models (2), which map the delayed vector to a scalar quantity, the neural difference (4) characterizes the evolution of the state vector $\boldsymbol{u}(k)$ over time, leading to an improved prediction horizon [22]. Due to the discrete nature of difference equations, (4) admits only evenly-spaced time series as input, and the performance of the model depends sensitively on the resolution of the time series.

Recent advances in neural ordinary differential equations (ODEs) [25]–[27] have allowed for calibrating latent dynamical systems in the reconstructed state space. However, the canonical approach for learning neural ODEs from data uses the adjoint method and calls for numerical solvers. Depending on the selected schemes, numerical errors accumulate as one iterates over time, and ultimately affect the loss function. Moreover, in order to obtain the gradients of the loss function with respect to network parameters, one needs to solve the neural ODEs forward and the corresponding adjoint ODEs backward, at each iteration. This can be computationally prohibitive for complex network architectures.

In this work, a self-consistent method is proposed for efficient learning of a latent autonomous dynamical system

$$\frac{\mathrm{d}\boldsymbol{u}(t)}{\mathrm{d}t} = \boldsymbol{A}\boldsymbol{u}(t) + \boldsymbol{e}(t) \quad \text{with} \quad \boldsymbol{A} \in \mathbb{R}^{m\times m}, \tag{5}$$

from noisy, incomplete time series, where $\boldsymbol{A} = \boldsymbol{A}(\boldsymbol{u})$ denotes a nonlinear matrix function of the state vector $\boldsymbol{u}(t)$. With $\boldsymbol{A}$ being approximated using a deep neural network to be discussed later, the matrix form (5) is as general as (3). We parameterize the discrete time series measurements by another neural network, denoted by $u(t)$, see Section II-A, and map it by the embedding function $\mathcal{E}(\cdot)$ to an m-dimensional state vector $\boldsymbol{u}(t)$ in the reconstructed state space, see Section II-B

$$\boldsymbol{u}(t) = \mathcal{E}[u(t)] \quad \text{with} \quad \mathcal{E}(\cdot) : \mathbb{R} \to \mathbb{R}^m. \tag{6}$$

To find the proper embedding dimension, denoted by $d \leq m$, we implement the FNN-Algorithm as an attention map to $\boldsymbol{u}(t)$, decomposing the m-dimensional state vector into d active components, and $m - d$ inactive components. Moreover, being a continuous and differentiable function of time, the parameterized state vector $\boldsymbol{u}(t)$ allows us to evaluate the deviation vector $\boldsymbol{e}(t)$ at each time, see Section II-C, avoiding a numerical integration of (5), and corresponding adjoint equations, during the training. Then, by requiring $\boldsymbol{u}(t)$ to be a solution to the latent dynamical system, we filter out additive noises that are not described by (5), signifying a self-consistency. As an ablation study, we consider the state space reconstruction using both the delay embedding and the autoencoder. On a synthetic time series, which is sampled from the Lorenz attractor [28] and masked by additive Gaussian noise, we demonstrate in Section III that the predictive power and the denoising feature of the proposed method mainly result from the self-consistency, at variance with previous studies. Finally, the limitations of the proposed scheme are discussed in Section IV, where conclusions are drawn.

II. Methods

A. Parameterization and Self-Consistent Filtering

Time series data are often unevenly spaced, and corrupted by noises. In order to obtain a continuous limit, we parameterize the time series $s_N = [s(1), s(2), \ldots, s(N)]$ measured at times $[t_1, t_2, \ldots, t_N]$ using neural networks

$$u(t) = \mathcal{N}_u(t; \boldsymbol{\theta}_u), \quad \mathcal{N}_u(\cdot) : \mathbb{R} \to \mathbb{R}. \tag{7}$$

The deviation loss associated with the parameterization

$$L_{\text{fit}} = \frac{1}{N\sigma_s^2} \sum_{k=1}^{N} \left[s(k) - \mathcal{N}_u(t_k; \boldsymbol{\theta}_u)\right]^2, \tag{8}$$

is normalized by the batch variance σ_s^2. Here, the index k is reserved for discrete measurements.

For small and noisy datasets, a minimization of L_{fit} with respect to $\boldsymbol{\theta}_u$ alone will overfit. Assuming that $\boldsymbol{u}(t)$, cf. (6), is both the true state underlying the time series, as well as a solution to the latent unknown dynamical system (5), we include the deviation $\boldsymbol{e}(t)$ as a regularizer

$$L_{\text{ode}} = \frac{1}{Md} \sum_{i=1}^{M} \sum_{j=1}^{m} \left[e_j(t_i)\right]^2. \tag{9}$$

As it shall be discussed in Section II-B, the proper embedding dimension, $d \leq m$, is learned during the training and implemented through an binary attention map to the state vector. Since our implementation yields only d active dimensions out of the m-dimensional dynamical system, we normalize the loss function L_{ode} by d. At each iteration, we compute the

loss by uniformly sampling M points from the time interval $t_i \in [t_1 + (m-1)\tau, t_N]$ with $i = 1, \ldots, M$. A successive re-sampling covers the entire solution manifold of (5). We defer our discussion on the learning Algorithm for $\mathcal{E}(\cdot)$ and d to Section II-B and on the functional form of $\boldsymbol{e}(t)$ to Section II-C, respectively. Note that the error vector $\boldsymbol{e}(t) \in \mathbb{R}^m$ is an implicit function of $\boldsymbol{\theta}_u$, a joint minimization of

$$L_u = L_{\text{fit}} + \lambda_u L_{\text{ode}}, \tag{10}$$

where λ_u is a hyper-parameter, ensures the smoothness of the solution trajectory and filters out additive noise.

B. State Space Reconstruction

Whitney's embedding theorem states that a d_F-dimensional attractor underlying the measured time series can be embedded in an m-dimensional delayed vector space, provided that $m \geq 2d_F$. This theorem, however, provides only an upper limit for the embedding dimension, a smaller value of m is usually sufficient [29]. While choosing too low a value of m makes the reconstructed dynamical system unable to resolve the complexity of s_N, successive redundancy associated with large values of m degrades the reconstructed dynamical system.

Instead of prescribing the embedding dimension a priori, we adopt Gilpin's idea [23] and start with a reasonably large value m for the configuration space, $\mathbb{R}^m$, where (5) lives, and search for an embedding subspace $\mathbb{R}^d$, with $d \leq m$, which contains the attractor. Given $u(t)$, the m-dimensional delay vector can be expressed as

$$\tilde{\boldsymbol{u}}(t)^T = [u(t), u(t-\tau), \ldots, u(t-(m-1)\tau)]. \tag{11}$$

To treat the redundancy associated with large values of m, we calculate the fraction of false nearest neighbors γ, a heuristic first proposed by Kennel *et al.* [24]. False nearest neighbors of a trajectory point in too low an embedding dimension will separate as the embedding dimension increases by one. This corresponds to a monotonic decreasing of γ with increasing dimensionality, until the proper embedding dimension d is reached. Therefore, the embedding dimension d can be inferred by examining how γ varies as a function of dimension. See Appendix for details.

Let $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_m]$, where γ_i denotes the fraction of false nearest neighbors associated with an i-dimensional embedding. Instead of incorporating $\boldsymbol{\gamma}$ into a loss function as in [23], we introduce a binary mask

$$\boldsymbol{w}^T = \lceil \text{relu}(\boldsymbol{\gamma} - \epsilon) \rceil = [\underbrace{1, \ldots, 1}_{d}, \underbrace{0, \ldots, 0}_{m-d}], \tag{12}$$

which decomposes the configuration space $\mathbb{R}^m$ into a d-dimensional embedding containing the attractor and $(m-d)$ redundant dimensions. Here, $\lceil \cdot \rceil$ denotes the ceiling function and $\text{relu}(x) = \max(x, 0)$. Following Kennel *et al.* [24], we take the threshold $\epsilon = 0.01$.

To assess the dependence of the proposed self-consistent method on how the state vector $\boldsymbol{u}(t)$ is recovered, we compare two approaches for the state space reconstruction; first, the method of delay, where the state vector reads:

$$\boldsymbol{u}(t) = \boldsymbol{w} \odot \tilde{\boldsymbol{u}}(t), \tag{13}$$

and second, the autoencoder

$$\text{Encoder}: \quad \boldsymbol{u}(t) = \boldsymbol{w} \odot \mathcal{N}_e[\tilde{\boldsymbol{u}}(t); \boldsymbol{\theta}_e], \tag{14a}$$

$$\text{Decoder}: \quad \hat{\boldsymbol{u}}(t) = \mathcal{N}_d[\boldsymbol{u}(t); \boldsymbol{\theta}_d], \tag{14b}$$

where $\odot$ denotes element-wise multiplication. The inclusion of $\boldsymbol{w}$ compresses the outputs of $\mathcal{N}_e(\cdot) : \mathbb{R}^m \to \mathbb{R}^m$ to the d-dimensional embedding, creating a bottleneck; while the decoder $\mathcal{N}_d(\cdot) : \mathbb{R}^m \to \mathbb{R}^m$ ensures information conservation. The associated reconstruction loss is

$$L_{\text{rec}} = \frac{d}{Mm \sum_{i=1}^{d} \sigma_{u_i}^2} \sum_{i=1}^{M} \sum_{j=1}^{m} [\hat{u}_j(t_i) - \tilde{u}_j(t_i)]^2, \tag{15}$$

where σ_{u_i} denotes the standard deviation of the i-th component of $\boldsymbol{u}(t)$ in the batch direction. To enforce an isotropic span of the attractor, we minimize the following loss function

$$L_{\text{exp}} = \frac{2}{d(d-1)} \sum_{i=1}^{d-1} \sum_{j=i+1}^{d} K_{ij}^2 + \frac{1}{d} \sum_{i=1}^{d} [\sigma_{u_i} - \text{mean}(\sigma_{\boldsymbol{u}})]^2, \tag{16}$$

where K_{ij} denotes the ij-th component of the covariance matrix $\boldsymbol{K} \in \mathbb{R}^{m \times m}$ of $\boldsymbol{u}(t)$. Thus, the encoder favors an orthogonal span, while the second term forces the unfolding to be isotropic. We include L_{ode} as a regularizer. With the weights $\lambda_{e,1}$ and $\lambda_{e,2}$ being hyper-parameters, the loss functions for the encoder and the decoder are:

$$L_e = L_{\text{rec}} + \lambda_{e,1} L_{\text{ode}} + \lambda_{e,2} L_{\text{exp}} \quad \text{and} \quad L_d = L_{\text{rec}}. \tag{17}$$

C. Neural Dynamical Systems

To confine the dynamics of (5) to the d-dimensional embedding, we introduce an FNN-informed attention $\boldsymbol{w} \cdot \boldsymbol{w}^T$ to the output matrix of a neural network $\mathcal{N}_f(\cdot) : \mathbb{R}^m \to \mathbb{R}^{m \times m}$

$$\boldsymbol{A} = (\boldsymbol{w} \cdot \boldsymbol{w}^T) \odot \mathcal{N}_f[\boldsymbol{u}(t); \boldsymbol{\theta}_f]. \tag{18}$$

Given $\boldsymbol{u}(t)$ defined in Section II-B, the deviation vector reads

$$\boldsymbol{e}(t) = \frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} u_1 \\ \vdots \\ u_d \\ 0 \\ \vdots \\ 0 \end{bmatrix} - \begin{bmatrix} A_{11} & \ldots & A_{1\,d} & 0 & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ A_{d1} & \ldots & A_{dd} & 0 & \ldots & 0 \\ 0 & \ldots & 0 & 0 & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & 0 & \ldots & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_d \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{19}$$

Here, A_{ij} denotes the ij-th output of $\boldsymbol{A}$ and the time derivatives are calculated using auto-differentiation [30].

The dimensionality reduction to an attractor is a common feature of dissipative dynamical systems [31]. For dissipative systems, the divergence of the vector field $\boldsymbol{F}(\boldsymbol{u}) = \boldsymbol{A}\boldsymbol{u}$ is negative. As we assume that there exists an attractor in the state space, we propose to penalize the positive divergence of $\boldsymbol{F}(\boldsymbol{u})$. Since the negative divergence is a global property of latent dynamical systems, we sample the configuration space uniformly with

Algorithm 1: Self-Consistent Learning of Neural Dynamical Systems.

Input: Mini-batched time series $\{s_N^1, s_N^2, \ldots, s_N^S\}$ and the corresponding time labels.
Guess initial parameters $\{\boldsymbol{\theta}_{u^1}, \boldsymbol{\theta}_{u^2}, \ldots, \boldsymbol{\theta}_{u^S}\}$, $\boldsymbol{\theta}_f$ and initialize $\boldsymbol{\gamma} = \boldsymbol{w}^T = [1, \ldots, 1]$.
Guess initial parameters $\boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_d$ if the embedding method is auto-encoder.

while not converged **do**
  Create an empty list: $\boldsymbol{U}$.
  for $s = 1, \ldots, S$ **do**
    Uniformly draw M samples $u^s(t_i)$ with $i = 1, \ldots, M$ and append to $\boldsymbol{U}$.
    Uniformly draw M samples from the configuration space†.
    Compute 3 losses: L_{fit}, L_{ode} and L_{div}.
    Optimize $\boldsymbol{\theta}_{u^s}$ using: $\partial_{\boldsymbol{\theta}_{u^s}}(L_{\text{fit}} + \lambda_u L_{\text{ode}})$.
    Optimize $\boldsymbol{\theta}_f$ using: $\partial_{\boldsymbol{\theta}_f}(L_{\text{ode}} + \lambda_f L_{\text{div}})$.
    if embedding method is auto-encoder **then**
      Compute 2 extra losses: L_{ref} and L_{exp}.
      Optimize $\boldsymbol{\theta}_e$ using: $\partial_{\boldsymbol{\theta}_e}(L_{\text{rec}} + \lambda_{e,1} L_{\text{ode}} + \lambda_{e,2} L_{\text{exp}})$.
      Optimize $\boldsymbol{\theta}_d$ using: $\partial_{\boldsymbol{\theta}_d} L_{\text{rec}}$.
    end if
  end for
  Compute epoch-wise $\hat{\boldsymbol{\gamma}}$ using $\boldsymbol{U}$, cf. Appendix, and update $\boldsymbol{\gamma}$ using a moving average: $\boldsymbol{\gamma} = (1-\alpha)\boldsymbol{\gamma} + \alpha\hat{\boldsymbol{\gamma}}$.
  Compute and update $\boldsymbol{w}^T$ using (12).
end while

Output: Optimized parameters $\{\boldsymbol{\theta}_{u^1}, \boldsymbol{\theta}_{u^2}, \ldots, \boldsymbol{\theta}_{u^S}\}$, $\boldsymbol{\theta}_f$ and weights $\boldsymbol{w}^T$, as well as $\boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_d$ if the auto-encoder is used.
Hyper-parameters used in this paper: $S = 128$, $M = 64$, $\lambda_u = \lambda_{e,1} = \lambda_{e,2} = \lambda_f = 1$ and $\alpha = 0.1$.
† For delay embedding, the configuration space is spanned by the delayed vectors that are scaled to $[-1, 1]$; while for auto-encoder, the configuration space is confined by the output activation function of the encoder, here $\tanh$.

M points, denoted by $\boldsymbol{u}_{r,i} \in \mathbb{R}^m$ with $i = 1, \ldots, M$, at each iteration. Then, the loss function reads

$$L_{\text{div}} = \frac{1}{Md} \sum_{i=1}^{M} \left\{ \text{relu}\left[\text{div} \boldsymbol{F}(\boldsymbol{u}_{r,i}) \right] \right\}^2 . \tag{20}$$

A joint minimization of L_{ode} and L_{div}

$$L_f = L_{\text{ode}} + \lambda_f L_{\text{div}}, \tag{21}$$

where λ_f is a hyper-parameter, enforces solution trajectories on the attractor.

D. Algorithm

The schematic Algorithm 1 brings together all pieces introduced in the previous sections. For neural dynamical systems whose state vector is reconstructed using the delay embedding (13), there are only three loss functions to jointly minimize. Components L_{fit} and L_{ode} relate to the self-consistent mechanism; a joint minimization yields the trajectory $\boldsymbol{u}(t)$ which underlies the observed time series measurements s_N and solves a latent continuous-time dynamical system (5), as well as the nonlinear matrix function $\boldsymbol{A}$ which is not known a priori. The third loss L_{div} results from the assumption that there exists an attractor in the state space. The assumption is generally applicable to any meaningful dataset in practice, as for dynamical systems with positive divergence, the corresponding time series would diverge. Since L_{div} is effective only when the divergence is positive, we observe that this loss function remains zero, with some small perturbations, for most of the time. Therefore, when using the delay embedding, the proposed method jointly minimizes essentially two losses L_{fit} and L_{ode}. The relative weight between these two losses reflects how confident we are in the measured data. In our case studies in Section III, we keep all hyper-parameters equal to 1, and we report that for any reasonable choice of weights ranging from $[0.1, 1]$, the results remain qualitatively unchanged.

The additional two losses, i.e. L_{rec} and L_{exp}, arise when the state space reconstruction is performed using the autoencoder. When constructing a latent dynamical system, one ensures that each coordinate of the state vector $\boldsymbol{u}(t)$ carries the maximum information by minimizing the correlation between them [29]. Geometrically, the decreased correlation among adjacent axes corresponds to an unfolding of the attractor in the state space. For the method of delay, the degree of unfolding is set by the delay time τ. For $\tau = 0$, the correlation among any two axes is 1, and the reconstructed attractor collapses to the diagonal line in the state space. With increasing values of τ, the attractor gradually unfolds from the diagonal line. For the autoencoder, the larger size of the attractor is due to the normalization σ_{u_i} in the reconstruction loss L_{rec}; while the unfolding from the diagonal line is enforced by the loss function L_{exp}. As discussed in detail in Section III, although the autoencoder can improve the generalization ability of the learned neural dynamical systems, the predictive and denoising features of the proposed method are mainly due to the self-consistency, at variance with previous works on the deep state space reconstruction [21]–[23].

For large dataset, Algorithm 1 allows dividing the training dataset into S batches $\{s_N^1, s_N^2, \ldots, s_N^S\}$ with an $(m-1)\tau$

overlapping in between, where the superscript $s = 1, \ldots, S$ indicates batch. Each batch is parameterized separately, leading to in total S parameterization functions $\{u^1(t), u^2(t) \ldots, u^S(t)\}$ that are jointly trained with one neural dynamical systems model. Hence, the computation time per epoch scales almost linearly with S. The scalability of our method allows for a distributed training of a long time series on multiple GPUs.

A Python implementation of Algorithm 1 with auto-encoder is provided as supplementary material. An implementation using the delay embedding can be recovered from the provided code by removing the autoencoder, accordingly.

III. Experiments With Synthetic Time Series

State space reconstruction [20]–[23] and the calibrated neural ODEs [25] are commonly assessed by visualizing the reconstructed attractor and by forecasting the continuation of the time series. In this work, we test the proposed method on the Lorenz equations [28]

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \sigma(y - x), \quad \frac{\mathrm{d}y}{\mathrm{d}t} = x(\rho - z) - y, \quad \frac{\mathrm{d}z}{\mathrm{d}t} = xy - \beta z, \tag{22}$$

with default values $\sigma = 10$, $\rho = 28$, $\beta = 8/3$, enabling a comparison with previous works. More specifically, we sample the x-variable of a trajectory generated by solving Eqs. (22) for 10,200 time steps from one initial condition $(x, y, z) = (0, 1, 1.05)$. The step size is taken to be $\Delta t = 0.05$. With η being a ratio, the noisy time series is prepared by corrupting the measurement $x(k)$ with a Gaussian white noise

$$s'(k) = x(k) + N(0, \sigma^2) \quad \text{with} \quad \sigma = \eta \sigma_x, \tag{23}$$

where σ_x denotes the standard deviation of sampled x-variable from the Lorenz attractor. A workflow for data preparation, self-consistent learning of neural dynamical system in delay coordinates, and demonstration on denoising and forecasting features of the proposed method is shown in Fig. 1.

In the model, we employ a residual connection [32] around two sub-layers made of 32 neurons that are regularized with batch normalization [33] and activated by a $\tanh$ function. Networks $\mathcal{N}_u, \mathcal{N}_e, \mathcal{N}_d$ and $\mathcal{N}_f$ consist of a stacking of 3 and 5 residual blocks, respectively. By scaling the noisy dataset $s'(k)$ to a range $[-1, 1]$, denoted now by $s(k)$, the output activation functions for $\mathcal{N}_u, \mathcal{N}_e, \mathcal{N}_d$ are $\tanh$. Since there is no a priori information on $\boldsymbol{A}$ (except for the divergence of $\boldsymbol{Au}$), we use a linear activation for the output of $\mathcal{N}_f$. For noisy dataset, we corrupt the input of the encoder with Gaussian noise $N(0, 0.5^2)$ as in [23], [34], [35] and apply a dropout regularization [36] with a rate 0.1 just before each network's output layer. A Python implementation of the proposed network architecture for the encoder can be found in supplementary material.

We separated the scaled time series $s(k) \in [-1, 1]$ into a training $t \in [0, 490]$, a validation $t \in [490, 500]$ and a test $t \in [500, 510]$ set; the training set was divided into 128 batches with an $(m-1)\tau$ overlapping between each batch. At each epoch, we call FNN-Algorithm detailed in Appendix to compute the fraction of nearest neighbors $\boldsymbol{\gamma}$ and substitute it into (12) for the mask $\boldsymbol{w}$. The FNN-Algorithm converged to and remained on a $d = 3$ embedding during the training, cf. Fig. 2. Without loss of generality, we selected $m = 6$. Following [29], we select $\tau = 2\Delta t = 0.1$ by visualizing the delayed attractor, as shown in Fig. 1(a). We trained our model using the ADAM optimizer [37] and on an NVIDIA 2080Ti GPU; iterating over an epoch took around 2 and 4 seconds without and with the autoencoder, respectively. Note that with $\mathcal{N}_f$ considered in this work the forward passing associated with a numerical integration of the neural dynamical system (5) over 100 and 500 units of time would take around 536 and 2,599 seconds, respectively, thus precluding a training using the previously proposed adjoint method [25]–[27]. We pretrained the models $\mathcal{N}_u$ and $\mathcal{N}_f$ with $\boldsymbol{u}(t)$ reconstructed using the delay embedding (13) for 15,000 epochs, and then turn on the autoencoder (14) for a fine tuning for 15,000 epochs. The results were compared with a direct training using the delay embedding for 30,000 epochs. During the training, we retained the models with the lowest normalized mean square error (NMSE) [18] on the validation set

$$\text{NMSE} = \frac{\sum_{k=1}^{N} (\text{truth}_k - \text{prediction}_k)^2}{\sum_{k=1}^{N} [\text{truth}_k - \text{mean}(\text{truth})]^2}. \tag{24}$$

TABLE I
Normalized Mean Square Error (NMSE) of Signals

Case	Raw measurements	$u(t)$	Decoder
$\eta = 0.00$	0.0	5.8×10^{-5}	9.6×10^{-5}
$\eta = 0.15$	2.3×10^{-2}	2.6×10^{-3}	2.6×10^{-3}
$\eta = 0.30$	9.1×10^{-2}	1.6×10^{-2}	1.3×10^{-2}

Column "Raw measurement" shows NMSEs of noisy time series computed with respect to the noise-free one. Column "$u(t)$" shows NMSEs computed using the parameterized time series without autoencoder; whereas column "Decoder" computes NMSEs using the output of the decoder.

A. Visualization of the Latent Attractor

Fig. 3 shows the reconstructed attractors from noisy measurements. Even for the case $\eta = 0.3$, the inclusion of the L_{ode} regularizer enables us to recover an attractor resembling that observed in noise-free conditions. Note that the denoising feature can come from two origins. First, the regularized parameterization removes, in principle, outliers that are not governed by (5). This features self-consistent filtering as the underlying dynamical system is not known a priori, but it is learned jointly with the parameterization function. Second, since the reconstruction loss, cf. (15), is not exactly zero at the end of the training, the deviation of the decoded signal from $u(t)$ may contribute to a denoising as reported in [21]–[23]. In order to assess the origin of denoising ability, we compare in Table I the NMSE of the noisy signal, the parameterization function $u(t)$, and the output of the decoder, all computed with respect to the noise-free Lorenz time series. Despite both deterministic chaos and noise being characterized by broadband spectra in the frequency domain, the proposed self-consistent method achieves an overall 80% noise reduction by separating noise from a chaotic signal, insensitive to the embedology.

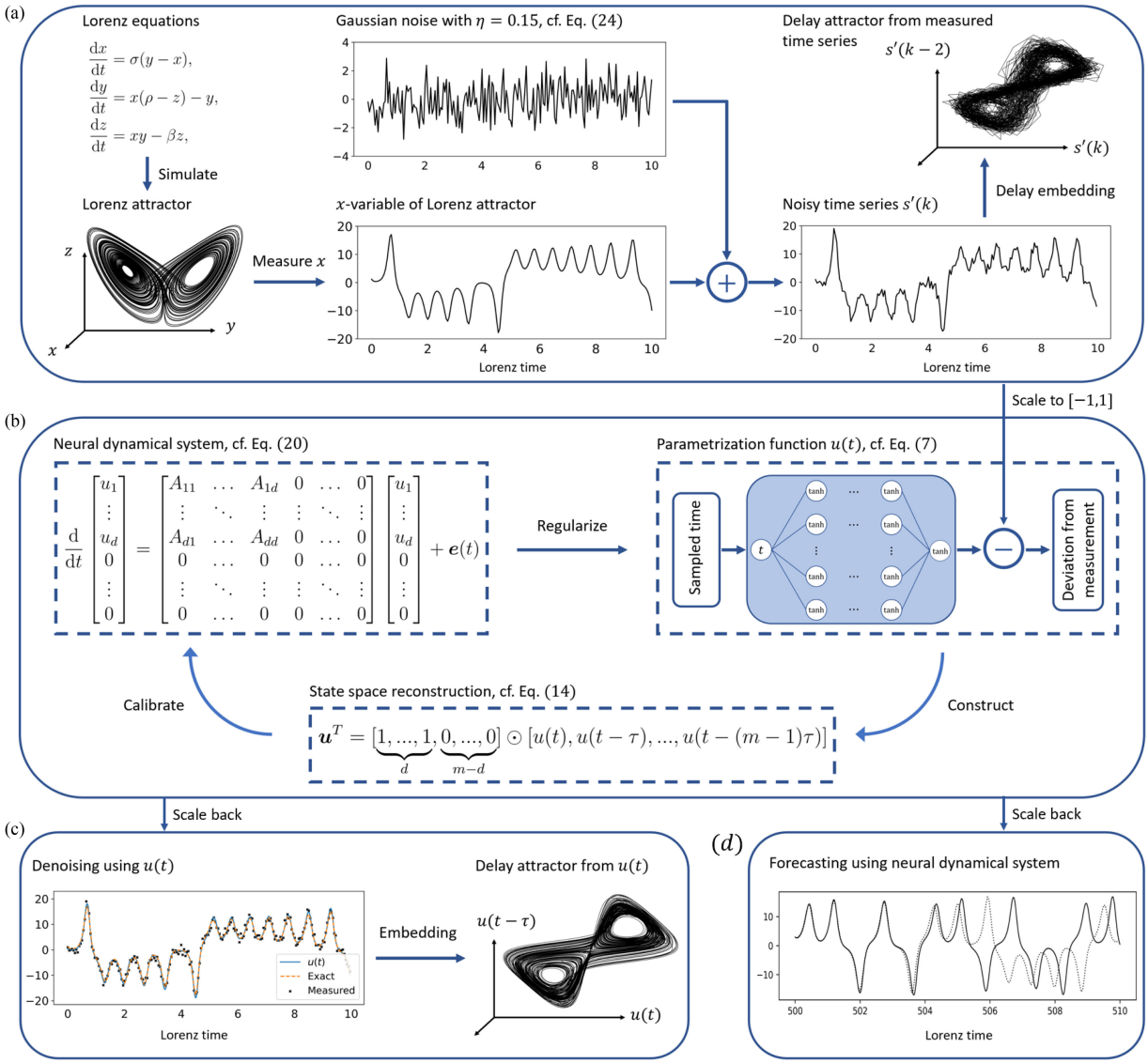

Fig. 1. Workflow of the numerical experiment. (a) Data preparation for time series sampled from the Lorenz attractor; (b) self-consistent learning of the neural dynamical system in delay coordinates; (c) denoising using the parameterized function $u(t)$; (d) forecasting by numerically integrating the calibrated neural dynamical system, where the solid curve denotes the predicted time series and the dashed one is the true time series.

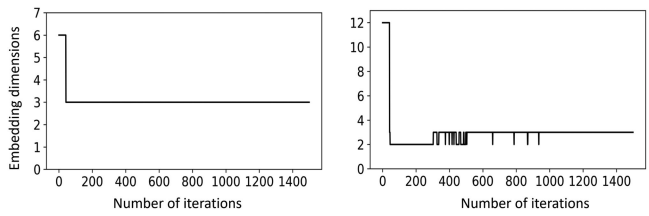

Fig. 2. Dimension $d = 3$ obtained from the FNN method for embedding in an $m = 6$ (left) and an $m = 12$ (right) dimensional configuration space within the first 1500 epochs.

B. *Continuation of the Training Time Series*

Two families of neural dynamical systems were identified: one in delay coordinates using the delay embedding (13), and the other in a latent space reconstructed using the autoencoder (14). To make a prediction using the trained neural dynamical systems, one needs to (i) find the initial condition $\boldsymbol{u}(t_0)$, (ii) numerically integrate the neural dynamical system from $\boldsymbol{u}(t_0)$ for a given period of time; and (iii) convert the solutions back to the measurement space. To get an initial condition, e.g. $t_0 = 500$, we parameterize the measured segment $t \in [499, 500]$ by minimizing L_u, cf. (10), and depending on the method of state space reconstruction, substitute the parameterized function to either (13) or (14a) for the state vector. The initial condition is recovered by evaluating the inferred state vector at $t = t_0$. To make a prediction, (5) is numerically integrated from the recovered initial condition $\boldsymbol{u}(t_0)$ forward in time and the solution is then converted to the measurement space. Upon a rescaling, the conversion reduces to an identity operation with the delay embedding; whereas, with the autoencoder, the conversion is made using the decoder (14b). Note that the parameters of the neural dynamical systems $\boldsymbol{\theta}_f$, as well as the autoencoder $\boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_d$, are fixed during the inference phase.

TABLE II
COMPARISON OF PREVIOUS METHODS ON PREDICTING THE CONTINUATION OF THE NOISE-FREE LORENZ TIME SERIES

	Details of the model	Prediction horizon
Autoregressive models	Local linear model in low-pass filtered delay coordinates. The training data is the x-variable of a Lorenz time series for $t \in [0, 500]$ with a time step $\Delta t = 0.05$ [20].	4.5
	LSTM with complete observation. The training data is x, y, z-variables of a Lorenz time series and their time derivatives for $t \in [0, 50]$ with a time step $\Delta t = 0.005$ [10].	3
Neural difference equations	Neural mapping function in a latent space. The training data is the x-variable of a Lorenz time series for $t \in [0, 100]$ with a time step $\Delta t = 0.01$ [22].	11
Neural ODEs	Adjoint method training in a latent space. The training data is the x-variable of a Lorenz time series for $t \in [0, 100]$ with a time step $\Delta t = 0.01$ [22], [25].	0.5
	Self-consistent learning in delay coordinates. The training data is the x-variable of a Lorenz time series for $t \in [0, 490]$ with a time step $\Delta t = 0.05$ (ours).	8.5
	Self-consistent learning in a latent space. The training data is the x-variable of a Lorenz time series for $t \in [0, 490]$ with a time step $\Delta t = 0.05$ (ours)	11.5

The prediction horizon is expressed in units of Lorenz time for comparison.

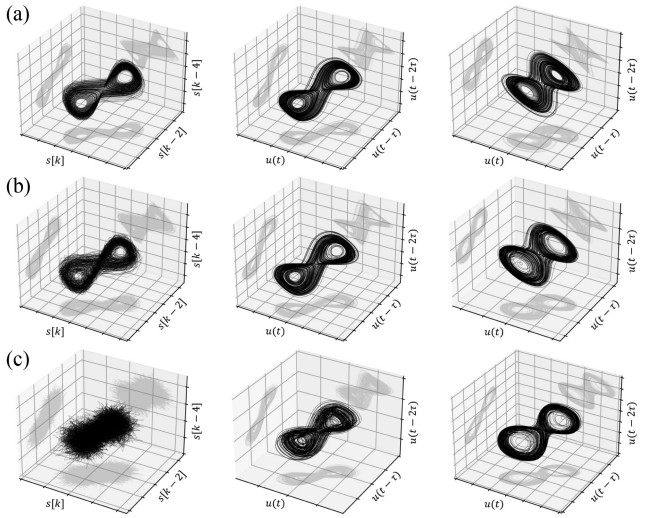

Fig. 3. Delay attractors reconstructed from noisy measurements (Left) and the parameterization function $u(t)$ (Middle). Right: Latent attractors reconstructed using the autoencoder. The time series were masked by Gaussian noise with noise ratio (a) $\eta = 0$; (b) $\eta = 0.15$; and (c) $\eta = 0.3$, cf. (23).

In Fig. 4, we predict the continuation of Lorenz time series for the period $t \in [500, 510]$, and compare the predictions with the exact solution of the Lorenz model. It is observed that, in the absence of noise, our approach allows for an accurate prediction of up to 8.5 and 11.5 units of time without and with the autoencoder. To avoid bias, we summarize previous results on forecasting the continuation of the Lorenz time series using both classical [20] and deep learning [10], [22], [25] methods in Table II. Note that predictions with the neural ODEs trained using the adjoint method diverge within 0.5 units of time, cf. Fig. 5 in [22]. Whereas in our case, even after the predicted trajectory deviates from the exact ones, instead of diverging, our prediction remains on the attractor. Therefore, the learned neural dynamical systems model is not only useful for short-term forecasting, but also for long-term simulation. The inclusion of noise leads to a deterioration in the prediction horizon, as expected. With increasing noise ratio, our method fails to find the precise initial conditions, see Fig. 4(c). The sensitive dependence on initial conditions of chaotic dynamical systems prohibits forecasting. Fig. 5 shows continuations of the noise-free training dataset $t \in [0, 490]$ into the future. Besides degradation of performance, the prediction horizons obtained with and without the autoencoder are comparable.

C. Forecasting From Different Initial Conditions

To assess the transferability of our prediction model, we study forecasting for time series initiated with different initial conditions from the training dataset. The Lorenz model (22) is integrated with randomly chosen initial conditions for the period $t \in [0, 110]$. With initial conditions $\boldsymbol{u}(t_0)$, where $t_0 = 100$, inferred from the interval $t \in [99, 100]$, we predict the continuation of the time series for $t \in [100, 110]$. The results are shown in Fig. 6. Compared with direct continuations (Figs. 4 and 5), the deterioration in prediction horizon suggests that the reconstructed dynamical system overfits to the training dataset, which is seeded with a single initial condition. Therefore, should a calibration of the general dynamical system be the goal, one needs to consider a training dataset with time series seeded from different initial conditions. Nevertheless, the use of autoencoder seems to enable a better generalization of the calibrated neural dynamical system. This, however, comes with the cost of doubled computation time.

IV. DISCUSSION AND CONCLUSION

We have introduced a general method which self-consistently allows us to filter a noisy time series without a priori information, to reconstruct an attractor from the filtered results, and to calibrate the underlying latent dynamical system. The measured time series is parameterized using deep neural networks. The latter being continuous functions of time, we augment the dataset by uniformly sampling the time interval and substituting into the parameterization function $u(t)$. The state space reconstruction is performed using either the delay embedding, which maps $u(t)$ to delay coordinates, or an autoencoder, which maps $u(t)$ to a latent space. The appropriate embedding dimension, which is not known a priori, is automatically searched using the FNN-Algorithm during the training and implemented as a novel attention mechanism to the state vector, as well as to the neural dynamical system. Finally, the latent dynamical system,

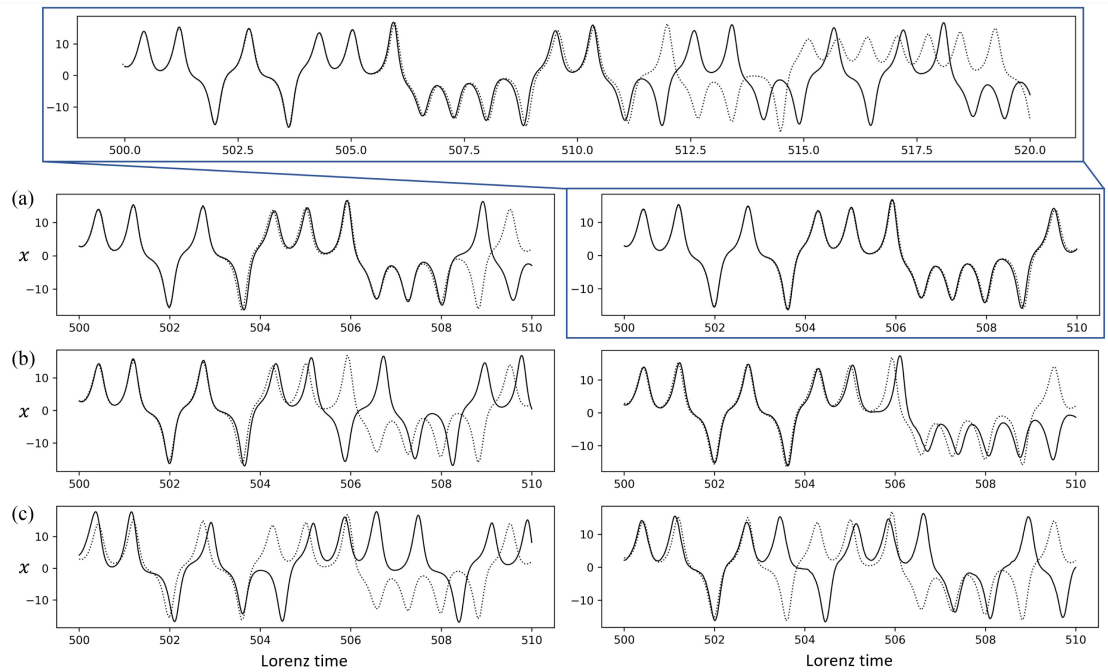

Fig. 4. Time series prediction for x-variable of the Lorenz attractor masked by an additive Gaussian noise with (a) $\eta = 0$; (b) $\eta = 0.15$; and (c) $\eta = 0.3$ without (left) and with (right) the autoencoder. The inset reveals that with the autoencoder and a clean training dataset, the prediction (solid) stays near the true signal (dashed) for around 11.5 units of Lorenz time.

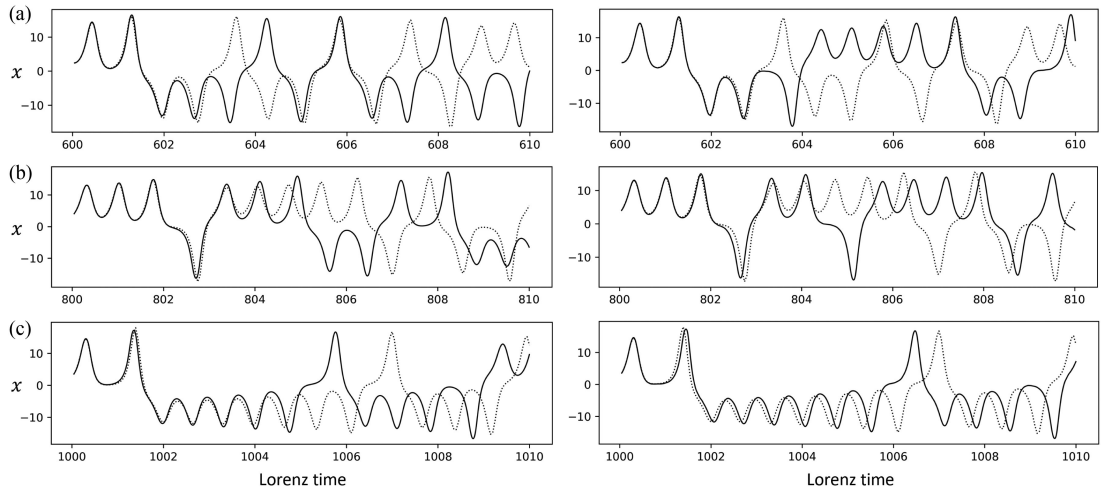

Fig. 5. Prediction for three successive continuations of the training dataset $t \in [0, 490]$ further into the future without (left) and with (right) the autoencoder (a) $t \in [600, 610]$; (b) $t \in [800, 810]$; and (c) $t \in [1000, 1010]$. The solid curve denotes the predicted time series and the dashed one is the true time series.

which underlies the temporal evolution of the state vector, serves as a regularizer for $u(t)$ to filter out the noise and for the encoder to determine an optimal embedding, completing the self-consistent cycle.

The proposed self-consistent method has been tested by forecasting the continuation of an univariate time series sampled from the Lorenz attractor. To the best of our knowledge, our prediction horizon is significantly longer than most published ones. Moreover, with an additive Gaussian noise, an overall 80% noise reduction is achieved for a chaotic signal. While we adopted the same architecture for all neural networks and kept the regularizer strength equal to one, recent studies [38]–[40] have showed that tuning the regularizers' strength and incorporating a priori knowledge, e.g. symmetries and leading order terms, can speed up the convergence and improve the quality of the learned model to a large degree. Thus, we trust that our results can still be improved.

The fundamental assumption underlying this work is that the time series is governed by some unknown ordinary differential equations. Therefore, our Algorithm is only applicable to deterministic time series masked by additive noise. Being aware that most time series of practical interest, e.g. financial data, are stochastic, one needs to replace (5) by stochastic differential equations and reformulate the loss functions.

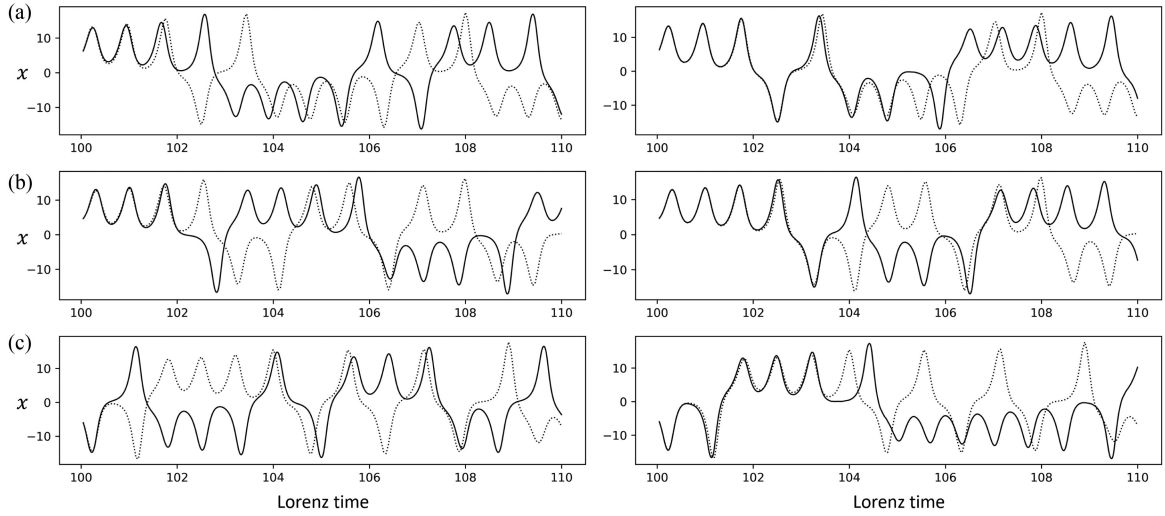

Fig. 6. Prediction ($t \in [100, 110]$) for Lorenz time series seeded with different initial conditions ($t = 0$) from the training dataset (a) $(x, y, z) = (1.02, 0.05, -1.67)$; (b) $(x, y, z) = (3.14, -1.59, 2.65)$; and (c) $(x, y, z) = (2.00, 3.00, 4.25)$ without (left) and with (right) the autoencoder. The solid curve denotes the predicted time series and the dashed one is the true time series.

Insofar as the time series is not largely stochastic, the present approach should be efficient to uncover the driving dynamical system. Once the calibration is done, it is possible to integrate the neural dynamical system to forecast the nearest future. Given the margins of reliability and robustness of the method, we hope to see that the proposed self-consistent method being applied as an auxiliary tool to explore hidden physics from experimental data and to extract relevant information from numerical simulations.

APPENDIX
FRACTION OF FALSE NEAREST NEIGHBORS

Consider a random sampling of M points $[t_1, \ldots, t_M]$ from a given time interval and substituting into (11) leads to

$$\boldsymbol{U} = \begin{bmatrix} u(t_1) & \ldots & u(t_1 - (m-1)\tau) \\ \vdots & \ddots & \vdots \\ u(t_M) & \ldots & u(t_M - (m-1)\tau) \end{bmatrix}. \tag{25}$$

Let us reorganize (25) into the following form: $\boldsymbol{H} = [\boldsymbol{H}_1, \ldots, \boldsymbol{H}_m]$, wherein the d-th component of $\boldsymbol{H}$ reads

$$\boldsymbol{H}_d = \begin{bmatrix} u(t_1) & \ldots & u(t_1 - (d-1)\tau) & 0 & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & 0 \\ u(t_M) & \ldots & u(t_M - (d-1)\tau) & 0 & \ldots & 0 \end{bmatrix}. \tag{26}$$

The Euclidean length of each delay vectors in $\boldsymbol{H}_d$ is

$$\boldsymbol{L}_d = \left[\sum_{j=1}^{d} \left(H_d(i,j) \right)^2 \right]^{1/2} \in \mathbb{R}^{M \times 1}, \tag{27}$$

where $H_d(i,j)$ denotes the ij-th component in the tensor $\boldsymbol{H}_d$. Using $\boldsymbol{H}_d$ and $\boldsymbol{L}_d$, the Euclidean distances between each pairs of two vectors in $\boldsymbol{H}_d$ can be expressed by the following symmetric matrix

$$\boldsymbol{D}_d = \left[\boldsymbol{L}_d^2 + \left(\boldsymbol{L}_d^2 \right)^T - 2\boldsymbol{H}_d \cdot \boldsymbol{H}_d^T \right]^{1/2} \in \mathbb{R}^{M \times M}, \tag{28}$$

where the broadcasting rule is used. Collecting the nearest neighbors, i.e. the smallest off-diagonal elements, along the horizontal direction

$$R_d(i) = \min \left(D_d(i,j) \right) \quad \text{for} \quad j > i, \tag{29}$$

the FNN-Algorithm considers the element i as a false nearest neighbor if either of the following two tests fails. They are

$$\left[\frac{R_d(i)^2 - R_{d-1}(i)^2}{R_d(i)^2} \right]^{1/2} > R_{\text{tol}}, \tag{30a}$$

$$\frac{R_d(i)}{R_A} > A_{\text{tol}}, \tag{30b}$$

for $d \geq 2$, where $R_A^2 = \frac{1}{M} \sum_{i=1}^{M} [u(t_i) - \text{mean}[u(t)]]^2$. We adopt the recommended values $R_{\text{tol}} = 10$ and $A_{\text{tol}} = 2$ for the thresholds in [24]. Let

$$\gamma_d(i) = \begin{cases} 1, & \text{if Eqs. (30a) or (30b) or d = 1 is true,} \\ 0, & \text{otherwise,} \end{cases} \tag{31}$$

the fraction of the false nearest neighbors associated with a d-dimensional embedding is

$$\gamma_d = \frac{1}{M} \sum_{i=1}^{M} \gamma_d(i). \tag{32}$$

Computing the fraction of false nearest neighbors associated with each sub-dimension $d \leq m$ leads to $\boldsymbol{\gamma} = [\gamma_1, \gamma_2, \ldots, \gamma_m]$.

REFERENCES

[1] G. U. Yule, "VII, on a method of investigating periodicities disturbed series, with special reference to Wolfer's sunspot numbers," *Philos. Trans. Roy. Soc. London. Ser. A*, vol. 226, pp. 267–298, 1927.

[2] A. Lapedes and R. Farber, "Nonlinear signal processing using neural networks: Prediction and system modelling," Los Alamos Nat. Lab., New Mexico, USA, Tech. Rep. LA-UR-87-2662; CONF-8706130-4, 1987.

[3] A. S. Weigend, B. A. Huberman, and D. E. Rumelhart, "Predicting the future: A connectionist approach," *Int. J. Neural Syst.*, vol. 1, no. 03, pp. 193–209, 1990.

[4] A. S. Weigend, "Connectionist architectures for time series prediction of dynamical systems," Ph.D. dissertation, Stanford Univ., Stanford, CA, USA, 1991.

[5] E. A. Wan, "Time series prediction by using a connectionist network with internal delay lines," in *Proc. Time Ser. Prediction Forecasting Future Understanding Past*, 1994, pp. 195–217.

[6] E. W. Saad, D. V. Prokhorov, and D. C. Wunsch, "Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks," *IEEE Trans. Neural Netw.*, vol. 9, no. 6, pp. 1456–1470, Nov. 1998.

[7] A. Borovykh, S. Bohte, and C. W. Oosterlee, "Conditional time series forecasting with convolutional neural networks," *J. Comput. Finance*, vol. 22, no. 4, 2017.

[8] F. A. Gers, D. Eck, and J. Schmidhuber, "Applying LSTM to Time Series Predictable Through Time-Window Approaches," in *Proc. Int. Conf. Artif. Neural Netw.*, 2002, pp. 193–200.

[9] P. Mirowski and Y. LeCun, "Dynamic factor graphs for time series modeling," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2009, pp. 128–143.

[10] P. Dubois, T. Gomez, L. Planckaert, and L. Perret, "Data-driven predictions of the Lorenz system," *Physica D, Nonlinear Phenomena*, vol. 408, 2020, Art. no. 132495.

[11] S. Li *et al.*, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 5244–5254.

[12] B. Lim, N. Loeff, S. Arik, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *Int. J. Forecasting*, vol. 37, no. 4, pp. 1748–1764, 2021.

[13] B. Lim and S. Zohren, "Time series forecasting with deep learning: A survey," *Phil. Trans. Roy Soc. A*, vol. 379, 2021, Art. no. 20200209.

[14] A. N. Gorban and D. C. Wunsch, "The general approximation theorem," in *Proc. Int. Joint Conf. Neural Netw.*, vol. 2, 1998, pp. 1271–1274.

[15] D. A. Winkler and T. C. Le, "Performance of deep and shallow neural networks, the universal approximation theorem, activity cliffs, and QSAR," *Mol. Informat.*, vol. 36, no. 1/2, 2017, Art. no. 1600118.

[16] H. Lin and S. Jegelka, "Resnet with one-neuron hidden layers is a universal approximator," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 6172–6181.

[17] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence*, Berlin, Heidelberg: Springer, 1981, pp. 366–381.

[18] A. S. Weigend and N. A. Gershenfeld, *Time Series Prediction: Forecasting the Future and Understanding the Past*. Reading, MA, USA: Addison-Wesley, 1994.

[19] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw, "Geometry from a time series," *Phys. Rev. Lett.*, vol. 45, no. 9, pp. 712–716, 1980.

[20] T. Sauer, "Time series prediction by using delay coordinate embedding," in *Proc. Time Ser. Prediction, Forecasting Future Understanding Past*, 1994, pp. 175–193.

[21] H. Jiang and H. He, "State space reconstruction from noisy nonlinear time series: An autoencoderbasedapproach," in *Proc. Int. Joint Conf. Neural Netw.*, 2017, pp. 3191–3198.

[22] S. Ouala *et al.*, "Learning latent dynamics for partially-observed chaotic systems," Chaos: *An Interdiscipl. J. Nonlinear Sci.*, vol. 30, no. 10, 2020, Art. no. 103121.

[23] W. Gilpin, "Deep reconstruction of strange attractors from time series," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 204–216.

[24] M. B. Kennel, R. Brown, and H. D. I. Abarbanel, "Determining embedding dimension for phase-space reconstruction using a geometrical construction," *Phys. Rev. A*, vol. 45, 1992, Art. no. 3403.

[25] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," *Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.

[26] I. Ayed, E. de Bézenac, A. Pajot, J. Brajard, and P. Gallinari, "Learning dynamical systems from partial observations," 2019, *arXiv:1902.11136*.

[27] C. Rackauckas *et al.*, "Universal differential equations for scientific machine learning," 2020, *arXiv:2001.04385*.

[28] E. N. Lorenz, "Deterministic nonperiodic flow," *J. Atmospheric Sci.*, vol. 20, no. 2, pp. 130–141, 1963.

[29] H. Kantz and T. Schreiber, *Nonlinear Time Series Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[30] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 5595–5637, 2017.

[31] R. Temam, *Infinite-Dimensional Dynamical Systems in Mechanics and Physics*. Berlin, Germany: Springer, 2012.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[33] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[34] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 1096–1103.

[35] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3372–2408, 2010.

[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[37] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–15.

[38] S. Wang, X. Yu, and P. Perdikaris, "When and Why Pinns Fail to Train: A Neural Tangent Kernel Perspective," *J. Comput. Phys*, vol. 449, p. 110768, 2021.

[39] S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient pathologies in physics-informed neural networks," *SIAM J. Sci. Comput.*, vol. 43, no. 5, pp. A3055–A3081, 2021.

[40] Z. Wang and C. Guet, "Deep learning in physics: A study of dielectric quasi-cubic particles in a uniform electric field," *IEEE Trans. Emerg. Top. Comput. Intell.*, to be published, doi: 10.1109/TETCI.2021.3086237.

Zhe Wang received the B.Eng degree in electrical engineering from Wuhan University, Wuhan, China, in 2013, the M.Sc. degree in power engineering, and the Ph.D. degree in mathematical physics from Nanyang Technological University, Singapore, in 2014 and 2019, respectively. He is currently a Research Fellow with CNRS@CREATE. His research interests include AI-enhanced modeling and solving for differential equations, fluid dynamics, and dynamical systems.

Claude Guet is currently a Visiting Professor with the School of Physical and Mathematical Science, Nanyang Technological University, Singapore. He has authored or coauthored more than 130 peer-reviewed papers with more than 6900 citations and an H-index of 42. His main research achievements include theoretical and experimental contributions to nuclear physics, atomic and plasma physics, and nanophysics. His research leading thread has been quantum many body physics and semi classical approximations. He is a knight in the French Ordre de la Legion d'Honneur and in the Ordre des Palmes Academiques.